Operative Traffic Management for Energy-Efficient Train Operation

Michael Ummels[1], Tilo Schumann[2]
German Aerospace Center (DLR)
Institute of Transportation Systems
Lilienthalplatz 7, 38108 Braunschweig, Germany
[1] E-mail: michael.ummels@dlr.de, Phone: +49 (0) 531 295-3425
[2] E-mail: tilo.schumann@dlr.de, Phone: +49 (0) 531 295-3506

Abstract
We present a system that performs a network-wide optimization of rail traffic with respect to energy consumption by informing trains of upcoming conflicts. Our system works in cooperation with a driver assistance system installed on board of trains but is not tied to a vendor-specific solution since it uses a vendor-independent protocol to communicate with the trains. In a simulated scenario, we were able to demonstrate a reduction in energy consumption by 45% as opposed to when our system was not in use.

Keywords
Railway optimization, Real-time traffic management, Energy efficiency, Dispatching systems, Driver assistance systems

1 Introduction

Traditionally, optimizing railway operation aims at improving punctuality or increasing line capacity. However, given rising energy costs, energy efficiency has become another important goal. Unfortunately, some of these goals are antagonistic. For instance, if a train travels at a lower speed than scheduled, it saves energy but this comes at the expense of punctuality. However, if there exists a sufficiently large recovery time or time allowance in the schedule, it is possible to save energy without sacrificing punctuality.

Recently, many train operators have installed a driver assistance system (DAS) into their cabs. Such a system assists the driver in maintaining a driving style that ensures both punctuality and energy efficiency (see Albrecht, 2008). The drawback of most DAS solutions is that they only optimize the trajectory of a single train and do not take into account disturbances by other trains. In particular, they might tell the driver to drive at maximum speed even if that entails a longer stop at a signal due to a conflict with a preceding train. If the DAS knew of the upcoming conflict, it could tell the driver to reduce its speed ahead of time in order to avoid an unscheduled stop entirely. However, in order to be aware of possible conflicts with other trains, the DAS needs to communicate with the traffic control center (TCC) where the dispatchers coordinate the traffic in their area. Example of such advanced systems are CATO (Lagos, 2001), which helps to avoid stops of the heavily loaded iron ore trains on the single track line from Kiruna to Narvik, the system employed by Swiss operator BLS (Achermann, 2013) in the Lötschberg base tunnel which helps the

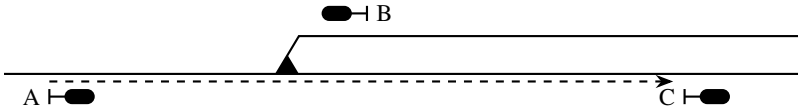

Figure 1: Route through an interlocking

trains to avoid a stop inside the tunnel before entering the single-track section, and the German system FreeFloat (Oetting and Glienicke, 2010), which has been tested in three of Deutsche Bahn's TCCs.

In contrast to these existing proprietary solutions, our system OVM (short for *Operatives Verkehrsmanagement*) is able to work with DAS systems from different manufacturers and is independent of the existing TCC infrastructure. OVM communicates in real-time with the DAS systems installed on the trains using the protocol EETROP (Wiebe and Sandor, 2010; see also Rackley, 2009), originally developed within the EU project *Railenergy*. Using EETROP, trains can send their exact position and speed, which allows for a much better conflict prediction than when train movement is only detected when a track section is cleared (as usual on most train lines). In the other direction, OVM uses EETROP to send *target points* to the trains, which indicate to the on-board DAS system when the train has to pass certain points on the track in order to avoid a conflict.

2 Theoretical Background

In this section, we recall some basic notions from the science of railway operation in order to formalize the problem solved by our system. As customary on most traditional lines, we assume a *fixed block system* where at any moment of time each block section may only be occupied by at most one train. A block section is usually guarded by a main signal which shows a stop aspect if the section is occupied and a distant signal placed in front of the main signal (possibly combined with the main signal for the preceding section), which signals the driver whether he can safely approach the section. Inside an interlocking, *routes* are used to protect the movement of a train: if several routes share a common infrastructure element such as a track or a switch, then at most one of the routes may be cleared for train movement at any moment of time. However, a route may be divided into several sections which may be released before the complete route is released (usually automatically using an axle counter) in order to establish another route which employs one of the released sections. Consider for instance the interlocking depicted in Figure 1. If a train uses the route from signal A to signal C, then the section from signal A to the switch should be released as soon as the train has cleared the switch in order to be able to set a route from signal B even when the section leading to signal C is still blocked.

It is important to note that a block or route section is not only occupied by a train when it is physically inside that section but also some time before and afterward. More precisely, Pachl (2008) calculates the *blocking duration* (also known as the *blocking time*) of a block section as the sum of the following durations:

1. the duration τ_{lock} for clearing the (main and distant) signals guarding the section,
2. the duration τ_{view} it takes for the driver to view the approach aspect on the distant signal,

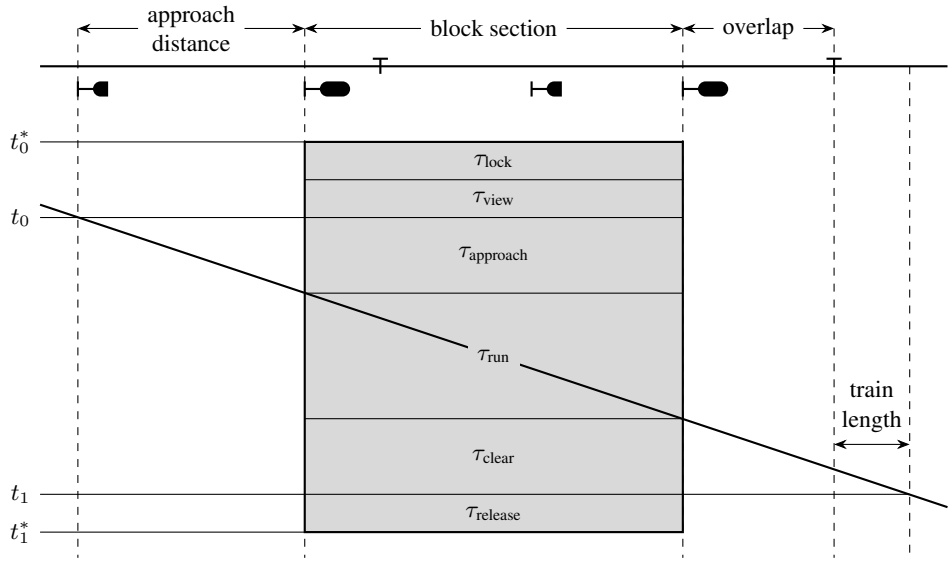

Figure 2: Calculating blocking windows

3. the approach duration τ_{approach}, i.e. the time the train takes to travel from the distant signal to the main signal,
4. the actual running time τ_{run} from the main signal to the next main signal,
5. the duration τ_{clear} to clear the block section with the full length of the train, possibly including a safety overlap after the next main signal, and
6. the release duration τ_{release}, i.e. the time for unlocking the section.

Hence, if a train passes the distant signal corresponding to a block section with its head at time t_0 and clears the block section with its full length at time t_1, the section is occupied from time $t_0^* \coloneqq t_0 - \tau_{\text{lock}} - \tau_{\text{view}}$ to time $t_1^* \coloneqq t_1 + \tau_{\text{release}}$. We call the interval $[t_0^*, t_1^*)$ the *blocking window* corresponding to the given train path and block section. See Figure 2 for a visualization of this calculation.

In case of a route inside an interlocking, the blocking window can be computed in a similar way. If the route is divided into separate sections, each section must be locked at once because there is only one signal that guards the whole route, but it may be released as soon as the train has passed the section with its full length. Hence, the blocking windows for the different sections of a route all have the same start time but a different end time; see Figure 3.

A *conflict* between two trains occurs if their blocking windows overlap for at least one block or route section. A typical example, depicted in Figure 4, is when a faster train follows a slower train so that the buffer between the two trains' blocking windows becomes smaller and smaller until a conflict occurs.

If the conflict is not prevented beforehand, this means that the trailing train will see a restrictive aspect at the distant signal guarding the section where the conflict occurs, which entails an unscheduled braking with negative consequences for energy consumption and line

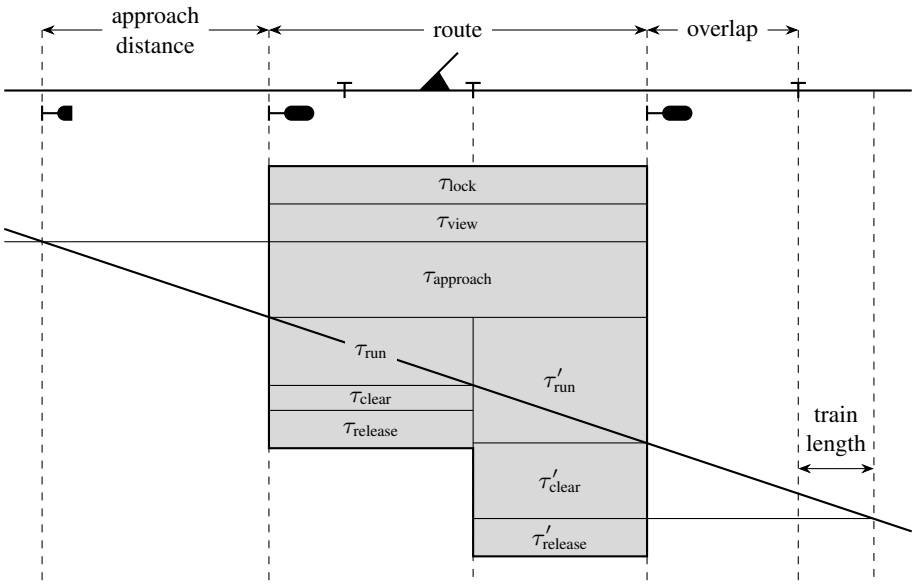

Figure 3: Blocking windows for a route with two sections

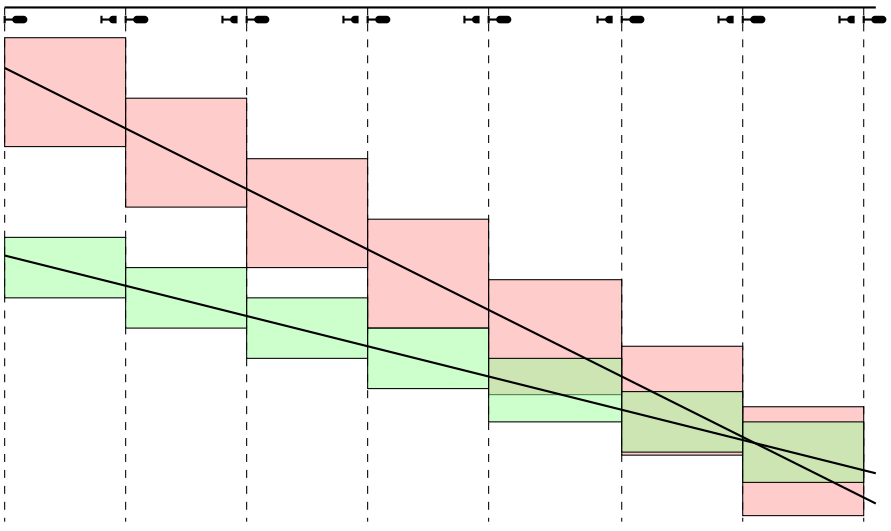

Figure 4: A conflict between two trains

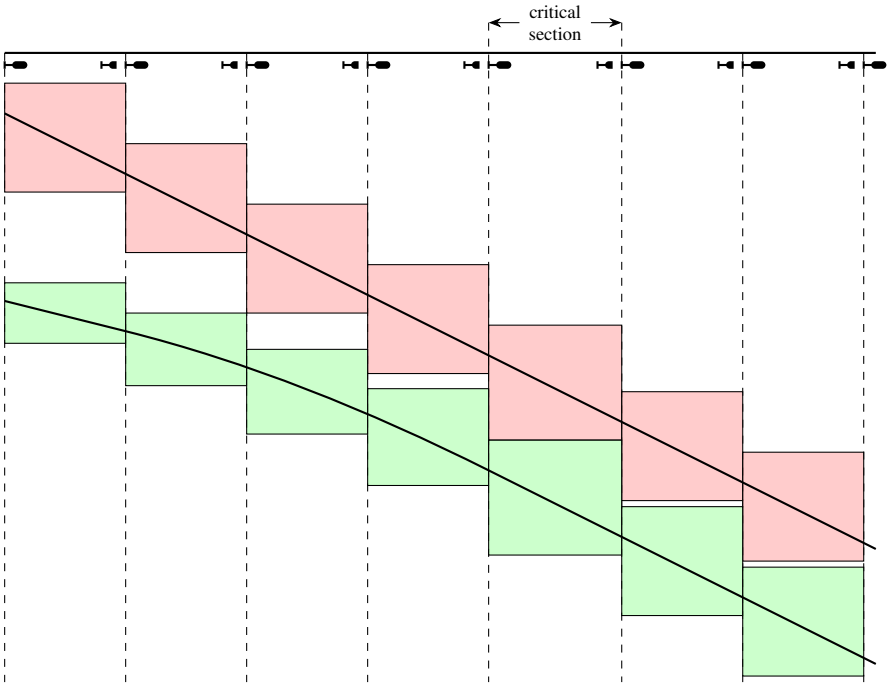

Figure 5: Conflict resolution

capacity. The easiest way to resolve the conflict is by adjusting the path of the trailing train so that the blocking windows do not overlap anymore but barely touch each other in at least one block section (the *critical section*); see Figure 5. In practice, this means that the train drives at a slower speed than calculated, but it will usually not come to a complete stop. In fact, the train might not even brake at all since a period of *coasting* (running without tractive effort) may be sufficient to attain the speed of the slower train ahead. Hence, if a conflict is resolved ahead of time instead of relying on the signaling system to resolve the conflict "by force", we expect less energy consumption and wear. Moreover since an unscheduled stop at a signal may also hinder other trains coming from behind, an early conflict resolution should also increase line capacity.

3 System architecture

Since OVM is not tied to a vendor specific DAS, OVM uses the published EETROP protocol to communicate with the on-board DAS units. The units are required to send the current position and speed of the train to OVM identifying themselves with their train numbers. In the opposite direction, OVM uses EETROP to communicate *target points* (ETPs) to the trains. EETROP distinguishes *mandatory* and *restrictive* ETPs. A mandatory ETP consists of a position on the track, a time instant, and a velocity, and has the semantics that the train should be at the given position at the given time with the given speed. A restrictive ETP consists of the same three elements but also indicates whether the given time and speed are *lower* or *upper* bounds. For instance, a restrictive ETP with position x, a time lower bound

of t, and a speed upper bound of v requires that the train passes position x no earlier than at time t and with a speed of at most v. The EETROP specification allows that several ETPs are sent inside one message, but exactly one of them must be a mandatory ETP. In practice, we can always use the next scheduled stop with the scheduled arrival time or earliest possible arrival time, whichever comes later, as the mandatory ETP and a restrictive ETP with a lower time bound and a trivial lower speed bound of 0 for each block section on the way to prevent conflicts (see the next section).

We would like to stress at this point that OVM does not send a complete detailed trajectory to the trains, but only as much ETPs as are necessary to avoid a conflict. Within the constraints enforced by these ETPs, the DAS is free to optimize the train's trajectory with respect to energy consumption or other criteria and present corresponding instructions to the train driver. The computationally heavy burden of computing an optimal trajectory is thus distributed to the DAS units on board of the trains, which do not need to take other trains into account when performing their optimization.

In order to compute the blocking windows, our system needs access to detailed infrastructure, timetable, and rolling stock data. For this purpose, OVM exposes an interface to *railML*®, an XML-based exchange format for railway data (see Huerlimann et al., 2004).

As mentioned in the introduction, our system works independently of the existing TTC infrastructure. This has the advantage that we do not have to provide a proof of safety for our system since it does not interact with the signaling system. However, this also means that all employed routes must be known to the system beforehand in order to anticipate conflicts. Moreover, in the case an alternative route is used, for instance to enable an unscheduled overtaking, OVM will become aware of the new situation only when the affected train has deviated from its original route. Hence, the system would certainly benefit from a standardized interface to connect to a TTC.

4 Implementation

We have implemented our system as a prototype in *Clojure*, a functional language for the *Java Virtual Machine*. As a prototype, the system works as a stand-alone GUI tool which communicates with the on-board DAS units using UDP as the network layer. After starting the tool, the user can import a railML® file with infrastructure, timetable, and rolling stock data. This data is written into an in-memory database, which allows for fast retrieval and querying.

Once all data has been imported, the user can put the tool into operational mode. In this mode, the tool repeatedly (e.g. every 30 seconds) runs Algorithm 1, which takes as inputs the position (and current speed) of each currently active train, computes approximations on their blocking windows, resolves conflicts, and outputs (restrictive) ETPs. The resulting ETPs together with the final mandatory ETP for the next stop are then sent to the trains, which can use these points to avoid a conflict.

Let us quickly go through the main steps in Algorithm 1. First, let us explain what we mean by the term *blocking window approximation* in step (1) of the algorithm: Since different DAS implementations may generate different driving advice and human drivers may interpret this advice differently, it is impossible to compute a train's trajectory and the corresponding blocking windows exactly. For this reason, we just compute an *over-approximation* on the blocking windows, i.e. for each section the computed interval should be a superset of the real blocking window. We achieve this by first computing the trajec-

Algorithm 1 Conflict detection and resolution

Input: Current train positions and speeds
Output: ETPs to be transmitted to the trains

for each *train*
  $\mathrm{ETPs}(\textit{train}) \coloneqq \emptyset$
  Compute blocking window approximations $B(\textit{train})$ (1)
Compute set C of upcoming conflicts (2)
while $C \neq \emptyset$
  let $c \in C$ be the first conflict in chronological order
  let $\textit{train}_1$ and $\textit{train}_2$ be the trains participating in c with $\textit{train}_1$ entering first
  Compute ETP e for $\textit{train}_2$ to resolve conflict (3)
  $\mathrm{ETPs}(\textit{train}_2) \coloneqq \mathrm{ETPs}(\textit{train}_2) \cup \{e_1\}$
  Recompute $B(\textit{train}_2)$ using $\mathrm{ETPs}(\textit{train}_2)$ (4)
  Recompute set C of upcoming conflicts (5)
return ETPs

tory with the shortest possible running time from the current position to the next stop (see Brünger and Dahlhaus, 2008), which establishes a *lower bound* on the future trajectory. We then compute an *upper bound* on the future trajectory by distributing a possible recovery time (time allowance), i.e. the difference between the computed earliest arrival time and the scheduled arrival time, over the computed trajectory and adding a fixed time buffer in order to cover for variances in departure times and driving style.

For both computed trajectory bounds, we compute the blocking windows as described in Section 2. As visualized in Figure 6, our over-approximation then takes for each block or route section the starting time from the blocking window corresponding to the lower bound and the end time from the blocking window corresponding to the upper bound. When compared with the probabilistic approach by Medossi et al. (2011), who defined a probability distribution on blocking windows, the set of blocking windows covered by our over-approximation should have a high probability in their model.

Given the blocking windows, we can easily compute the set of conflicts in (2) by checking for overlaps, where we only look for conflicts within a predefined time horizon, e.g. for the next 30 minutes. If no conflict exists, no ETP needs to be generated. Otherwise, the algorithm takes the first upcoming conflict and generates an ETP for the train that enters the conflict section last in (3). As shown in Figure 7, this ETP ensures that the train passes the distant signal ahead of the conflict section only after the other train has finished occupying the section and a duration of $\tau_{\text{lock}} + \tau_{\text{view}}$ has passed. In fact, to speed up the computation, in our implementation we do not only generate one ETP for the first conflict but also one for each conflict that follows the first one and involves the same two trains. This covers the most likely scenario of a fast train following a slow train where a conflict does not only involve one block or route section but several successive ones.

Since ETPs influence the trajectory of a train, we need to recompute the blocking windows for the train that has received new ETPs in step (4). While a lower bound on the future trajectory that obeys the given ETPs can be found easily, the challenge in establishing a tight upper bound is to find those ETPs that need to be met precisely (i.e. not later than at the given time) in order to minimize the delay at the next stop. As it turns out, this

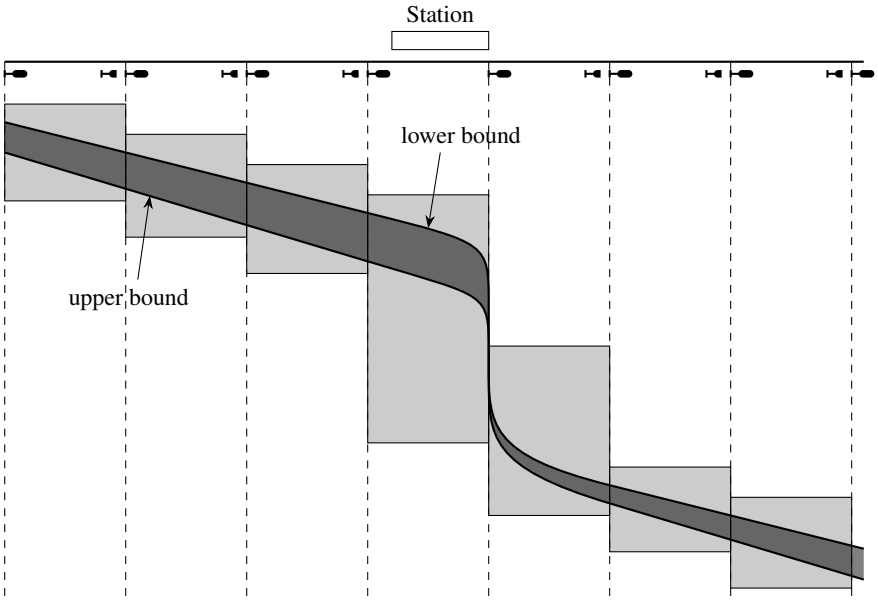

Figure 6: Blocking window approximations

problem is akin to the problem of determining the *convex hull* of a given set of points, which can be solved efficiently (see Cormen et al., 2009).

Finally, in step (5) we recompute the set C of upcoming conflicts. This is necessary because while we have solved one conflict by modifying the trajectory of one train, this modification might create new conflicts with other trains. However, progress towards a conflict-free operation has been made since any subsequent conflict will begin at a later time than the resolved conflict.

5 Demonstration

We have demonstrated our system in DLR's rail simulation environment RailSiTe® together with a DAS developed by Technical University Dresden and Interautomation Germany, on the double-track electrified main line from Paderborn to Warburg in Germany. For the test scenario, we selected this line because at 54 km it is sufficiently long and comes with a highly realistic visualization in the cab view. Being a double-track main line with several stations, it is also typical for a line where our system could be employed in real life.

Figure 8 shows how we have integrated OVM and DAS into the RailSiTe. Whereas in reality OVM would receive the trains' positions from the trains directly, we have adapted the RailSiTe to send the positions of all simulated trains to OVM. One of the simulated trains can be driven manually using the RailSet and this is where we installed the DAS, which also receives the train's current position and speed from the RailSiTe.

The test scenario consisted of a slower regional train operated automatically followed by an express train. Apart from Paderborn and Warburg, the regional train had scheduled stops at Altenbeken and Willebadessen, whereas the express train only had one intermediate

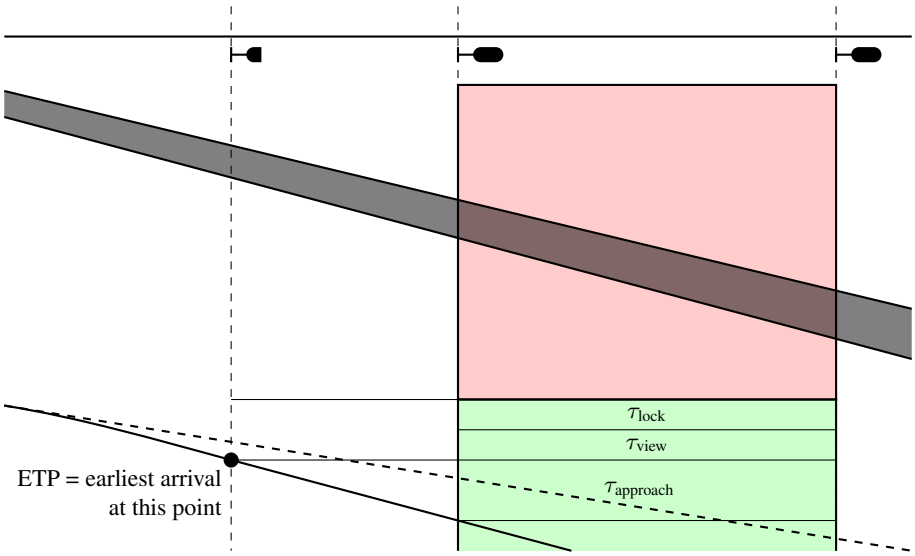

Figure 7: Computing an ETP

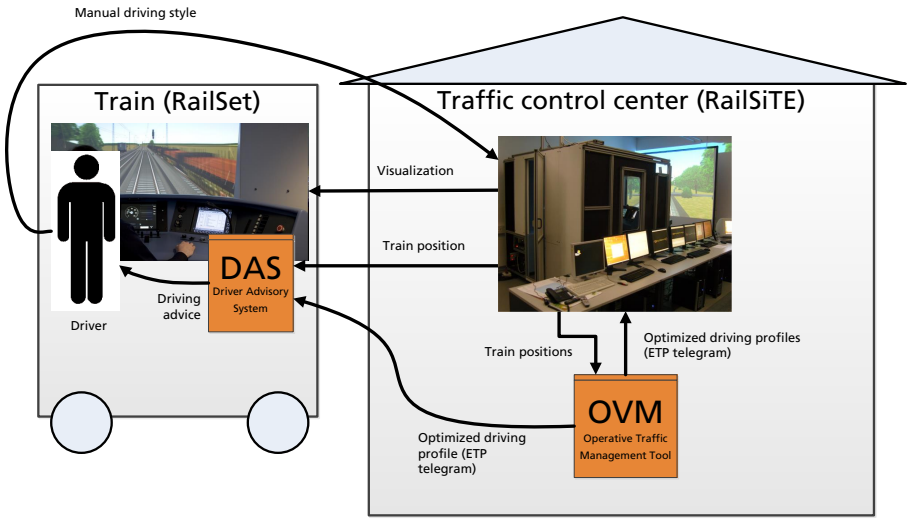

Figure 8: Integration of OVM and DAS into the DLR simulation environment

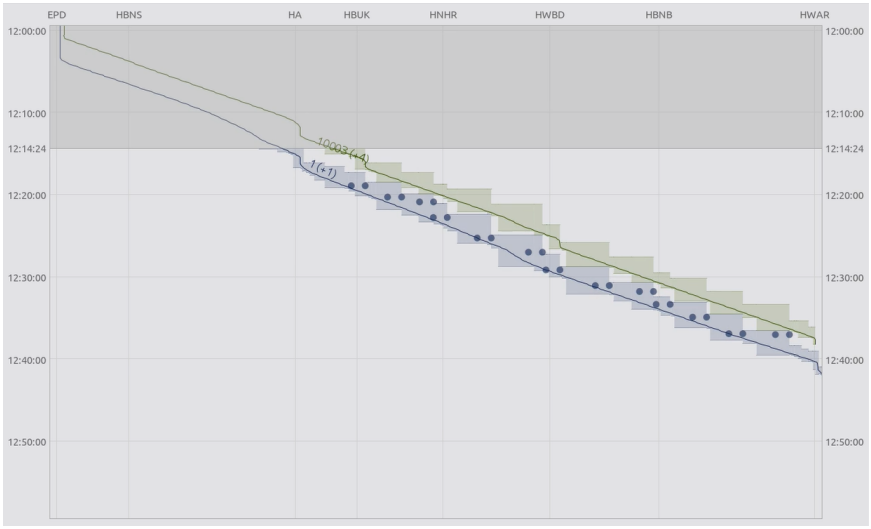

Figure 9: Screen dump of OVM in operation

scheduled stop at Altenbeken. The regional train was allowed to travel at a maximum speed of 100 km/h, whereas the express train was allowed to operate at a speed of 120 km/h. In the initial conflict-free schedule the regional train would have completed its run in Warburg just before the arrival of the express train, so in order to generate conflicts we let the regional train leave Paderborn with a delay of five minutes (which was reduced to four minutes on arrival in Warburg).

In order to evaluate the effects of the assistance, we executed the test scenario twice, first with OVM and DAS in operation and then without any assistance. Due to time constraints, we did not perform another test run with only the DAS switched on but OVM switched off. However, since the express train was delayed for most of its run, we expect that the DAS would not have helped in saving much energy in this case. Figure 9 shows a screen dump of OVM in operation during the test scenario (with colors inverted for better readability).

6 Experimental Results

Figure 10 shows the speed profiles of the regional train and the express train, both with OVM and DAS switched on and with OVM and DAS switched off. In both runs, the express train completed its run including the intermediate stop at Altenbeken in 39 min, which is 5 minutes longer than scheduled due to the conflict with the preceding regional train. As one can see in the diagram, the non-assisted driver had to brake and accelerate much more frequently than the assisted driver. In the non-assisted run it happened quite often that the driver saw a restrictive aspect on a distant signal so that he had to brake but could accelerate again once the train approached the main signal which had in the meantime switched to a proceed aspect. Moreover, the train came to a complete stop and hat to wait for some time at the home signal of Altenbeken and the station exit signal of Willebadessen since the section ahead was still occupied by the regional train ahead. Compared to the non-assisted run, the run where OVM and DAS was switched on went much smoother and the train driver never

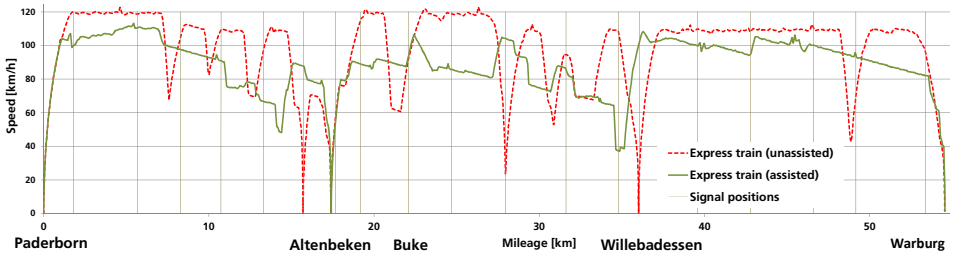

Figure 10: Speed profiles

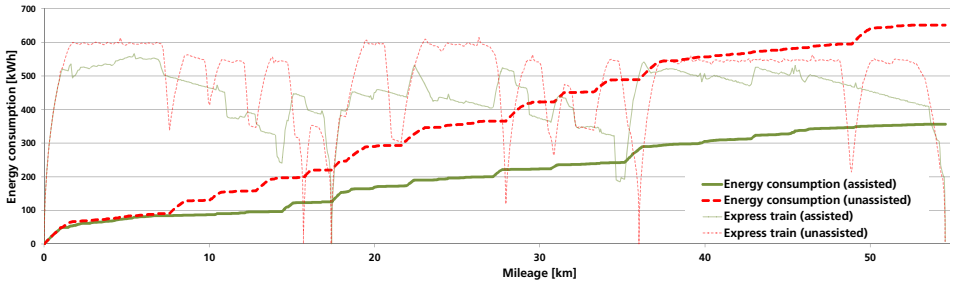

Figure 11: Energy consumption

observed a restrictive aspect at a distant signal.

Figure 11 shows the cumulative energy consumption (*work*) for the two runs of the express train, compared with the corresponding speed profiles. The assisted train consumed 356 kWh, whereas the non-assisted train consumed 651 kWh. Hence, energy consumption was reduced by 45% without sacrificing travel time. In the beginning of both runs, the train accelerated to near its maximum speed so that energy consumption was about the same. However, as soon as the non-assisted driver had to brake and accelerate again at the first distant signal showing a restrictive aspect, the curve for the non-assisted run quickly rises and then remains about double as high as the curve for the assisted run indicating an energy consumption about twice as high. Our results are summed up in Table 1.

7 Perspectives

While we could demonstrate the potential for a massive reduction in energy consumption, it should be noted that our tool does not yet anticipate dispatching decisions such as unscheduled overtakings. Only after such an overtaking has taken place, our system notices

Table 1: Summary of results

	Energy consumption [kWh]	Travel time [min]
OVM/DAS assisted run	356	39
Non-assisted run	651	39

that the train order has changed and adapts to the new situation. We therefore plan to include dispatching decisions into our prediction. In fact, since it is extremely hard to find the "best" dispatching decision automatically and an automatic tool can hardly replace the dispatcher's expert knowledge, we plan to evaluate different dispatching decisions, rank them and present them to the dispatcher so that she can select the one she favors.

We haven't focused on the benefit on line capacity in our studies, but we plan to study this effect in the near future. We believe that it is important to look at all the potential benefits of our system, because the focus on energy efficiency might be a declining trend. For instance, if there is a strict separation of train and network operation as advocated by European lawmakers, then there is no real incentive for the network operation company to install a system that helps train operators save energy. Moreover, modern electric locomotives are able to recuperate electrical power when braking and feed it back into the power network, for which they might get reimbursed. For instance, in Germany train operators receive up to 75% of the kWh price for every kWh they feed back (Bundesnetzagentur, 2013).

References

Achermann, E. (2013). Erfahrungen mit der neuen Verkehrsleittechnik im Lötschberg-Basistunnel. *ETR – Eisenbahntechnische Rundschau 10/13*, 72–75. In German.

Albrecht, T. (2008). Energy-efficient train operation. In I. A. Hansen and J. Pachl (Eds.), *Railway Timetable & Traffic*, pp. 83–105. Eurailpress.

Brünger, O. and E. Dahlhaus (2008). Running time estimation. In I. A. Hansen and J. Pachl (Eds.), *Railway Timetable and Traffic*, pp. 58–82. Eurailpress.

Bundesnetzagentur (2013). Marktuntersuchung Eisenbahnen 2013. In German.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

Huerlimann, D., A. Nash, V. P. Krauß, and J. Schütte (2004). railML® – A standard data interface for railroad applications. In *Proc. 4th International Conference on Railway Engineering Design and Optimisation (COMPRAIL 2004)*.

Lagos, M. (2001). CATO offers energy savings. *Railway Gazette International 5/11*, 50–52.

Medossi, G., G. Longo, and S. de Fabris (2011). A method for using stochastic blocking times to improve timetable planning. *Journal of Rail Transport Planning & Management 1*(1), 1–13.

Oetting, A. and J. Glienicke (2010). FreeFloat – Technologische Innovationen zur Steigerung der Kapazität im bestehenden Netz. *ETR – Eisenbahntechnische Rundschau 12/10*, 824–829. In German.

Pachl, J. (2008). Timetable design principles. In I. A. Hansen and J. Pachl (Eds.), *Railway Timetable & Traffic*, pp. 9–42. Eurailpress.

Rackley, S. (2009). Powercut. *The Rail Engineer 62*, 28–30.

Wiebe, E. and J. Sandor (2010). Railenergy brochure with an overview about the project and the final results. Publishable Final Activity Report NRG-UIC-D-7.1-192, UIC.